FNPL BPM System Readout Software

Jason Wennerberg
Fermi National Accelerator Laboratory

March 2, 2004

Abstract

This note describes the software created to facilitate the readout of the FNPL beam position monitor system on a Linux platform.

1 Introduction

The beamline of the A0 photoinjector at Fermilab is equipped with button beam position monitor (BPM) electromagnetic *pickup stations*. BPM *read-out electronics*, developed and manufactured at DESY, are used to acquire the beam position and orbit data. Figure 1 gives an impression of

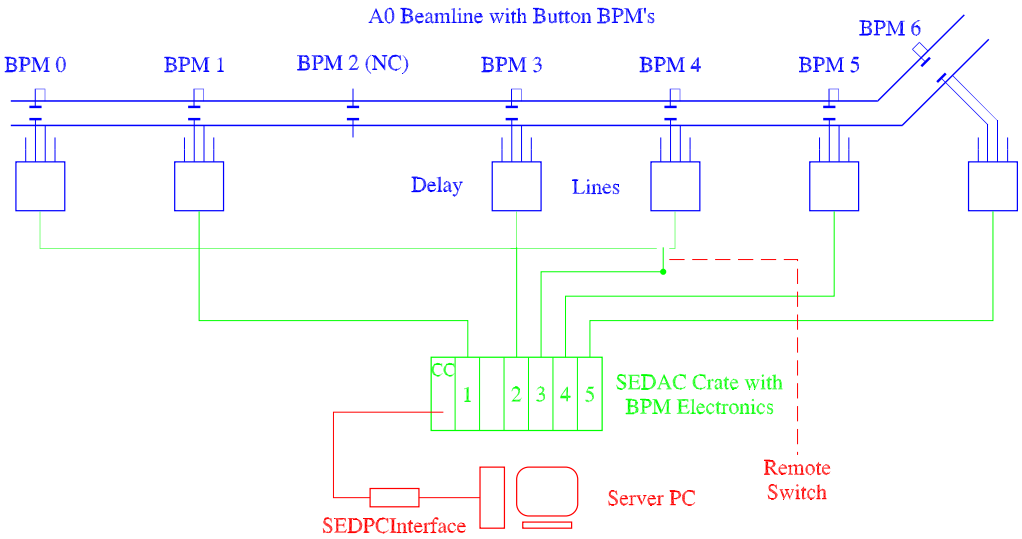

Figure 1: Overview of the BPM system of A0

the BPM hardware at A0. The stsyem now has 8 BPM's. Most of the BPM electronics hardware is concentrated in two 19″ SEDAC (*SE*rial *D*ata *AC*quision) crates. They keep all analogue and digital electronics, as well as a serial link to the server PC.

The existing software is injorb was written in Visual Basic for the Windows operating system. All software made for monitoring BPM's on the Linux OS are based on this software and a library written by Manfred Wendt of DESY called SedPc.

2 New software Operation

The new software is called oobpmread and was written December 2003 in the C language. It is run via the controll panel with the following switches available:

- -d: run and write debugging messages to screen
- -i: The number of reads to perform
- -f: file name for output
- -s: selects which BPM(1-8) to read
- -q: quiet mode: runs with no output to screen
- -a: used instead of -s to read all BPM's
- -w: sets the number of times to poll the DATA READY bit before giving up
- -I: do not initialize

For example, to read all the BPMs once and save it to a file called 'data.bpm' one would type 'oobpmread -a -i 1 -f data.bpm' in the terminal. To read a single BPM (BPM2) 10 times and write it to the file 'data.bpm2' one would type 'oobpmread -s 2 -i 10 -f data.bpm2'.

When the program is run with simply -a, the BPM's are initialized and ready to go. Single BPM's can then be read and each individual read will be written to screen and to the file as defined by -f. The statistical average computed over i reads will also be written to the screen.

A package that mimics the old software display was also created. It is called py_bpm_display.py and is obviously a Python script. This package interacts with the BPM's by making system calls to oobpmread and writing the data to files. It then uses these files to display the data in graphical form. This program has a display of position vs. time at the top of the display, with 8 buttons to select which BPM to read. It also has a two bar graphs which display the x and y values of each BPM for the current read. Other buttons are available to save the data, change the scales of the display and use a reference orbit. Figure 2 shows a screen shot.

3 New Software Overview

The procedure for reading all beam position monitors is outlined in the paper in reference [1]. Here is listed a short summary of the general procedure assuming you are starting from scratch and need to initialize the BPM's. In this case you must run oobpmread with the switch -a for 'all'.

1. The program is run via the command line with options
2. The BPMs are initialized by reading the information about their addresses, constants, etc. from a file
3. The Sed crate is initialized using the function 'SedIni' from SedPc

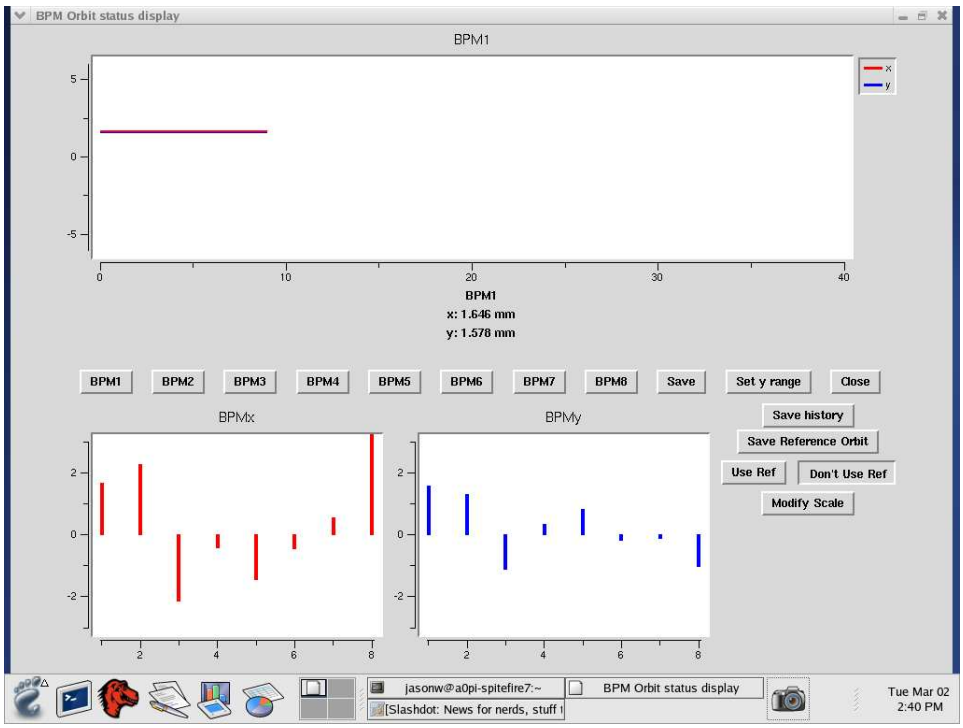

Figure 2: BPM Display monitor

4. Test for communication with the crate
5. Initialize the ADD module. The procedure for this is in table 5 of reference [1] and in section 6.1
6. ADC module is initialized and attenuation is set. This procedure is also listed in table 5 of reference [1] and in section 6.2
7. Clear overspeed bit
8. Clear register
9. Poll data ready bit on ADD to wait for DATA READY
10. Read data and check for saturation bit
11. Check to see if sum of E values (see reference [2]) is between min and max values (400 - 800)
12. If sum of E is not acceptable or saturated set bad attenuation flag for that BPM and go back to reinitialize the ADD and ADC and set new attenuation (increment by 1 or -1)
13. If sum of E is acceptable keep data and stop

4 Single Bunch Readout

There is a desire to be able to read and record single bunch readings in the BPM system. This is difficult only because of the need to set the individual attenuations of each of the BPM's. The new

software tentatively solves this problem by keeping a file up to date with the last known 'good' attenuation value. This is good as long as the beam does not change significantly in which case the program auto corrects however takes longer to read the devices. In order to read a single bpm oobpmread must first be run with the switch -a to initialize if the BPM's need to be initialized. It is a good idea to initialize the BPM's if the beam has been off since the last time they were read out. After the initialization single BPM's can be read using the -s switch in oobpmread.

5 Remote Operation

Remote monitoring of the BPM system is possible by conecting to a0pi-spitefire6.fnal.gov via telnet with a CryptoCard. If you don't have an account on a0pi-spitefire6.fnal.gov that matches a kerberos principal, something else will have to be worked out.
Once connected to the system the program oobpmread can be found in /home/jasonw/BPM/originals and run as discussed in section 2.
The program py_bpm_display.py can be found and run in directory /home/jasonw by typing 'python2.3 py_bpm_display.py &' as long as you have an X client running.

6 Software Details

Table 1 outlines the commands needed to initialize the BPM modules. They are explained further in sections 6.1 and 6.2.

module	register	action	value	comment
ADD	BSA+ 7	write	0	switch to SEDAC OPERATION
ADD	BSA+12	write	0	reset RELEASE MODE
ADD	BSA+10	write	0	switch to PROGRAM-CONTROLLED RELEASE MODE
ADD	BSA+ 9	write	254	set ADD = 255-254 = 1
ADD	BSA+ 8	write	0	set CLEAR & START
ADD	BSA+15	write	0	clear OVERSPEED
ADD	BSA+13	write	0	set RELEASE MODE
ADC	BSA+ 3	write	0	reset TRIGGER-FLAG
ADC	BSA+ 1	write	1	set TRIGGER = INTERNAL
ADC	BSA+ 2	write	0	reset OVERLOAD-FLAG
ADC	BSA+ 0	write	n	set ATTENUATOR = n dB (n = 0...63)
ADD	BSA+15	write	0	clear OVERSPEED

Table 1: Initialization commands.

Tables 2 and 3 give further information on the ADC and ADD modules.

register	action	value	comment
BSA+0	write	n	set ATTENUATOR = n dB (n = 0...63)
BSA+1	write	n	n = 0: set TRIGGER = GATE (int. trigger with gate) n = 1: set TRIGGER = INTERNAL n = 2: set TRIGGER = EXTERNAL
BSA+2	write	0	reset OVERLOAD-FLAG
BSA+3	write	0	reset TRIGGER-FLAG
		Q 0: H	1 dB
		Q 1: H	2 dB
		Q 2: H	4 dB
		Q 3: H	8 dB
		Q 4: H	16 dB
		Q 5: H	32 dB
		Q 6: ! H	INTERNAL-TRIGGER-FLAG set by an int. trigger event
BSA+ 0		Q 7: H	EXTERNAL-TRIGGER-FLAG set by an ext. trigger event
		Q 8: H	TEMPERATURE $> 40^0$ C
		Q 9: H	TEMPERATURE $> 50^0$ C
		Q10: H	TEMPERATURE $> 60^0$ C
		Q11: H	TEMPERATURE $> 70^0$ C
		Q13: H	OVERLOAD-FLAG, attenuation set to 63 dB
		Q14: L	GATE-MODE
		Q15: H	TRIGGER-MODE: EXTERNAL
		Q15: L	TRIGGER-MODE: INTERNAL

Table 2: Command reference of the ADC-module.

register	action	value	comment
BSA+ 6	write	0	switch to μP OPERATION
BSA+ 7	write	0	switch to SEDAC OPERATION
BSA+ 8	write	0	set CLEAR & START
BSA+ 9	write	n	set AD D = 255-n (# of additions as 1st-complement value)
BSA+10	write	0	switch to PROGRAM-CONTROLLED RELEASE MODE
BSA+11	write	0	switch to EXTERNAL RELEASE MODE
BSA+12	write	0	reset RELEASE MODE
BSA+13	write	0	set RELEASE MODE
BSA+15	write	0	clear OVERSPEED & test INPUT-DATA
BSA+10	read	Q 4: H	EXTERNAL RELEASE MODE active
		Q 5: H	CONNECTED
		Q 6: H	μP-MODE
		Q 6: L	SEDAC-MODE
		Q 7: H	DATA READY
BSA+12	read	Q0...13	DATA left BPM-electrode(4)
		Q14: H	OVERSPEED left BPM-electrode(4)
		Q15: H	OVERLOAD left BPM-electrode(4)
BSA+13	read	Q0...13	DATA right BPM-electrode(3)
		Q14: H	OVERSPEED right BPM-electrode(3)
		Q15: H	OVERLOAD right BPM-electrode(3)
BSA+14	read	Q0...13	DATA down BPM-electrode(2)
		Q14: H	OVERSPEED down BPM-electrode(2)
		Q15: H	OVERLOAD down BPM-electrode(2)
BSA+15	read	Q0...13	DATA up BPM-electrode(1)
		Q14: H	OVERSPEED up BPM-electrode(1)
		Q15: H	OVERLOAD up BPM-electrode(1)

Table 3: Command reference of the ADD-module.

6.1 Initializing the ADD module

Initialization of the ADD module is performed as follows where BSA refers to the Base address on the ADD:

1. Set SEDAC OPERATION ON by writting 0 to BSA+7
2. Reset release by writting 0 to BSA+12
3. Set program controlled release by writting 0 to BSA+10
4. Set ADD = 1st complement value so that 255 = 255-255 = 0 by writting 255 to BSA+9
5. Set Clear by writting 0 to BSA+8
6. Set release by writting 0 to BSA+13
7. Reset OVER-SPEED and test INPUT-DATA by writting 0 to BSA+15

There is a 9000 microsecond programmed delay in between each write for the hardware to keep up.

6.2 Initializing the ADC and setting the attenuation

Initialization of the ADC module is performed as follows where BSA refers to the Base address on the ADC:

1. Reset trigger flag by writting 0 to BSA+3
2. Set trigger to internal by writting 1 to BSA+1
3. Reset overload flag by writting 0 to BSA+2
4. Set attenuation by writting attenuation value to BSA+0
5. Read back attenuation and compare for good measure

6.3 Data Acquisition

Data aqusition is performed by looping over the commands in table 4

register	action	value	comment
BSA+ 8	write	0	set CLEAR & START
BSA+10	read	mask Q7	repeat until Q7 = H (DATA READY)
BSA+15	read	Q0...Q13: up BPM-electrode(1) Q14:	acquire data
BSA+14	read	Q0...Q13: down BPM-electrode(2) OVERSPEED	
BSA+13	read	Q0...Q13: right BPM-electrode(3) Q15:	
BSA+12	read	Q0...Q13: left BPM-electrode(4) OVERFLOW	

Table 4: Acquire the measured data of the ADD-module.

6.4 Data Post-processing

Data post-processing is required to compute the normalized horizontal and vertical beam displacement Δ_h and Δ_v from the four acquired signal values of the electrodes E_i:

$$\begin{aligned} \Delta_h &= k_h \frac{\tilde{E}_4 - \tilde{E}_3}{\tilde{E}_4 + \tilde{E}_3} \\ \Delta_v &= k_v \frac{\tilde{E}_1 - \tilde{E}_2}{\tilde{E}_1 + \tilde{E}_2} \end{aligned} \tag{1}$$

The calibration factor (so-called *monitor constant*) (k_h, k_v) varies for different geometries of the BPM pickup. $\tilde{E}_i$ in eqn. (1) are the modified mean-values of the acquired electrode signals E_k: The mean-value is computed according to the ADD-values n, set at the BSA+9 registers of the ADD-modules.

$$\bar{E}_i = \frac{E_i}{n}$$

This mean-value $\bar{E}_i$ has to be modified for two aspects, the difference of the cable attenuation in the delay-line and the nonlinearities of the electronics at lower levels:

$$\tilde{E}_i = k_{\mathrm{att}_i} \bar{E}_i + 10$$

First order compensation of the nonlinearities is done by simply adding a "10" to each of the four digital values.

All the constants $k_h, k_v, k_{\mathrm{att}_i}, ...$ are kept in a file that is read into the program during the initialization process.

References

[1] K. Desler, J. Neugebauer, R. Neumann, F. Wedtstein, and M. Wendt, "Programming the A0 BPM System," Not published.

[2] K. Desler, R Neumann, and M. Wendt, "Introduction to the A0 BPM System," Not published.